Towards a Maturity Model for Empirical Studies of Software Testing

Mark Harman,
Robert Hierons,
Goldsmiths College,
University Of London,
New Cross,
London SE14 6NW, UK.
Tel: +44 (0)171 919 7860
Fax: +44 (0)171 919 7853
M.Harman@gold.ac.uk

Mike Holcombe,
The University of Sheffield
Regent Court,
211 Portobello Street
Sheffield, S1 4DP, UK.
Tel: +44 (0) 114 222 1800
Fax: +44 (0) 114 222 1810
m.holcombe@dcs.shef.ac.uk

Bryan Jones,
University of Glamorgan,
Pontypridd,
CF37 1DL, UK.
Tel: +44 (0)1443 482730
Fax: +44 (0)1443 482715
bfjones@glam.ac.uk

Stuart Reid,
Cranfield University,
RMCS Shrivenham,
Swindon SN6 8LA, UK.
Tel: +44 (0)1793 785490
Fax: +44 (0)1793 785366
Stuart.Reid@rmcs.cranfield.ac.uk

Marc Roper,
Strathclyde University,
Livingstone Tower,
26 Richmond Street,
Glasgow G1 1XH, UK.
Tel: +44 (0)141 548 2956
Fax: +44 (0)141 552 5330
marc@cs.strath.ac.uk

Martin Woodward,
University of Liverpool,
Chadwick Building,
Peach Street,
Liverpool L69 7ZF, UK.
Tel: +44 (0)151 794 3676
Fax: +44 (0)151 794 3715
M.R.Woodward@csc.liv.ac.uk

July 20, 1999

Abstract

Software testing is an increasingly important part of the software development process. There is a growing realization that measurement and testing, though imperfect, offer perhaps the best approach to ensuring software safety and quality.

Unfortunately, there is no conclusive theoretical foundation to guide the practice of software testing and those empirical results which have been reported in the literature often appear to conflict.

Though a great deal more empirical evidence is required to facilitate a better understanding of the differences between software testing approaches, it is important to avoid an overwhelming avalanche of incomparable evidence. It is important to establish standards for the conduct of empirical testing studies in order that the results may be compared and that the lessons drawn more widely exploited.

This position paper is an attempt to create discussion, with the aim of formulating the first draft of a 'maturity model' for empirical studies of software testing.

It appears that the approach (or an improved version of it) may be applicable to empirical work in general, as well as to empirical assessment of testing (which is our primary concern here).

1 Introduction

Testing is a dominant feature of maintenance activity, because software surgery is often both preceded and proceeded by testing.

There exists a bewildering array of possible testing techniques [3]. Sadly, the theoretical foundations of the subject are not yet firmly established. The results of several theoretical studies appear to conflict [8, 19, 17, 9]. Moreover, empirical evidence appears to lead to differing conclusions [15, 7, 2, 16, 18, 10, 5].

One recent contribution to the emerging debate on software testing, has been the formulation of standards [14]. These have gone a long way to setting out the landscape of testing research and application. However, standards do not and should not prescribe a 'best method'. Because of the diversity of the field, empirical study is essential in order to establish the relative merits and most suitable application areas for different test techniques.

This paper sets out a basis of a framework into which empirical studies of testing might be located. The aim of the framework is that empirical studies should be comparable as far as possible. The framework must, however, accept that one single standard approach to experimentation would be over prescriptive and this led us to the concept of a *level* based approach inspired by (but rather different to) the SEI Capability Maturity Model [13].

The model attempts to avoid a situation in which empirical testing work is conducted in isolation, producing results which are hard to interpret and translate into different domains.

It is in the nature of the paper that it raises more questions than it answers. Specifically, unanswered but important questions are:

- What information should be presented concerning the system under test, its development process and the faults within it?
- How can a set of universally accepted benchmark systems and program fragments be established for the evaluation of testing strategies?
- How should the approach cater for the evolution of these data requirements and the addition of new benchmarks?
- How can replication and dissemination of the results of empirical studies be effectively supported and encouraged?

2 Support for Replication and Dissemination

The lack of past empirical work, suggests that an upsurge in empirical study of software testing is desirable. However, such an upsurge of interest and activity will bring with it fresh problems for the software testing community. It will be important to ensure that independent efforts are made to replicate results (to improve confidence in them) and to disseminate results to practitioners and other research workers.

A cultural shift in the software engineering research community is required. Currently empirical study appears to be viewed by many in the wider software engineering community as a rather dull and tiresome activity, offering little opportunity for creativity. This perception is not shared by business and industry which cries out for empirical results from independent agencies. Clearly the academic community should be taking the lead here. There are encouraging signs that a shift in perception towards a recognition of the value of empirical studies is taking place.

However, even if this cultural shift continues, there remains the problem of ensuring that results are replicated. In other scientific and engineering disciplines, empirical study and replication of results is part of the very fabric of the subject. Replication of results is seen as a key factor in determining confidence. In the software engineering paradigm this need for confidence through replication could not be greater. The astonishing diversity of software development methods, tools, technologies, users, staff expertise and products increases the volume of evidence required to achieve confidence.

3 A Capability Maturity Model for Empirical Studies of Software Testing

The goals of the empirical testing maturity model we propose are:

- To **facilitate comparison** between different empirical testing studies.
  This should be possible for all studies conducted at level three or above, because all processes used will be defined. Some, more limited, comparison will also be possible at level two.
- To **encourage replication** studies.
  Any study conducted at level two or above will offer the possibility of replication. This should encourage authors to construct experiments which meet the requirements of this level[1]. It would also, hopefully, encourage replication because a potential replicator will not have to waste time considering whether or not a candidate study is suitable for a replication effort[2].
- To **encourage adoption** of the model itself.
  Any model of testing which seeks to unite work and facilitate comparison and replication of results must aim to achieve a wide uptake in the testing and empirical communities. We hope that the graduated and inclusive nature of the model will allow researchers to adopt the model as a framework with relative ease.

Our initial attempt at defining a suitable level-based model is based loosely upon the CMM capability maturity model [13]. It is interesting to see the, perhaps surprising, closeness of fit between the CMM (designed to assess maturity of organisations) and our model (designed to assess maturity, and thus reusability and applicability, of experiments). However, perhaps our attempt to follow the CMM so closely is a little over-simplistic.

Level 1 An experiment is at level one, the **initial** level, if it provides insufficient information to allow any

[1] We would not claim to have fully worked out these requirements in the present paper, which is merely intended to initiate a discussion.

[2] Ideally, a central resource (for instance a web site) listing such studies would be provided by the testing community.

form of replication. Such an experiment is likely to be small scale.

Level 2 An experiment is at level two, the **repeatable** level, if it offers some information to allow an attempt at replication, though not all processes will be well-defined. Such an experiment is still likely to be small scale.

Level 3 An experiment is at level three, the **defined** level, if sufficient information is available to allow replication. In such an experiment, all processes will be well defined, though the experiment may still be conducted on a small scale.

Level 4 An experiment is at level four, the **large scale** level, if it is defined and applied to a large scale system. The large scale application of the experiment increases the likelihood that predictions can be inferred from the results reported.

Level 5 An experiment is at level five, the **optimizing** level, if it produces new approaches to experimentation which would improve future experiments. 'New approaches' might include new benchmark systems and examples or theoretical measures of test effectiveness.

Clearly more work is required to ensure that this model captures the consensus of the empirical software engineering community. It is possible that the maturity of an experiment should be assessed on several orthogonal dimensions, raising interesting issues of comparability and ordering between experiments.

It is also possible that the model (or an improved version of it) should and could be applied to more than just experiments (but also to case studies, for example).

Also, although we have focussed our attention on software testing, there appears to be no reason not to attempt a similar level based (perhaps multi-dimensional) model for the more general class of empirical studies of software engineering.

4 Related Work

In [12] it is argued that more experimental work in software testing is necessary in order to place testing techniques onto a scale of measurement, or classify them in such a way that is useful to the software engineer. Current experimental practices are examined using a parametric framework and are shown to contribute little towards a cohesive and useful body of knowledge. The idea of a benchmark repository of faulty and correct software is explored enabling unification of diverse experimental results. Such a unification should start the process of moving towards an evaluation taxonomy of testing methods.

There have been two proposals for frameworks to support the reporting of software engineering experiments, namely those of Lott and Rombach[11] and Basili, Selby and Hutchens[1].

Basili et al. [1] propose a framework to

> "... help structure the experimental process and to provide a classification scheme for understanding and evaluating experimental studies."

Their scheme consists of four categories which correspond to the standard phases of the experimental process - 1) definition, 2) planning, 3) operation, and 4) interpretation. They provide a tabular representation of the framework which refines each of the four categories - for example, definition is decomposed into the following elements: motivation, object, purpose, perspective, domain, and scope. Each of these elements are then assigned attributes which together capture a structured definition of the experiment.

Lott and Rombach [11] have proposed a "characterization scheme" for experiments that evaluate source code verification and validation techniques (although it would appear that the scheme is generalisable to any class of software engineering experiment). This scheme

> ' ... permits the comparison of results from similar experiments and establishes a context for cross-experiment analyses."

Although the Lott/Rombach scheme was not explicitly derived from that of Basili et al there is some similarity. They are both based on the four phases of standard experimental design, and the Lott/Rombach framework is basically enforcing structure for standard experimental design, containing the four major categories of: Goal Hypotheses and Theories, Experimental Plan, Experimental Procedures, and Results. Each of these are then broken down further into sub-categories to ensure a complete and consistent reporting mechanism.

Both Lott and Rombach and Basili, Selby and Hutchens essentially propose standards for the conduct of experiments. The novelty of our approach is that we introduce the notion of levels, thereby avoiding the prescription of a single standard, replacing it with a 'graduated set of standards'. We believe that this will encourage uptake, replication and evolution of experimental investigation.

Ericson, Subotic and Ursing [4] and Gelperin [6] introduce maturity models for improving the testing process. Like the present work, these models are inspired by the SEI Capability Maturity Model. Unlike the approach advocated here, these two models are aimed at improving an organisational processes (like the CMM), rather than providing a scale of maturity for empirical work.

The former [4] uses a four level model while the latter [6] uses a three level model. Ericson et al. argue that the increased number of levels is advantageous because it provides for a smoother transition between levels, encouraging participation. If the granularity is too large, then improvement is inhibited, because there is a perception that insufficient progress is being made.

This observation may have implications for the maturity model for empirical studies of software testing advocated here. Clearly, a model should not have too many levels to be workable, but the model should be inclusive and encourage process improvement.

5 Conclusions and Future Work

This paper introduces a maturity model for empirical studies of software testing. The details of the model remain to be worked out. The purpose of the paper is to initiate discussion of how to

- **encourage** more empirical study of software testing in the wider software engineering community,
- ensure that the results of such work will be **comparable** and
- encourage and **facilitate replication** of this work.

Although we have focussed our attention on software testing (our primary concern), there appears to be no reason not to attempt a similar level based (perhaps multi-dimensional) model for all empirical work in software engineering.

More work is required to extend and reformulate the model we propose here. We believe that our single-dimension approach may be over-simplistic; a multi-dimensional approach may be more appropriate.

References

[1] Basili, V., Selby, R., and Hutchens, D. Experimentation in software engineering. *IEEE Transactions on Software Engineering SE-12*, 7 (1986), 733–743.

[2] Basili, V. R., and Selby, R. W. Comparing the effectiveness of software testing strategies. *IEEE Transactions on Software Engineering SE-13*, 12 (1987), 1278–1296.

[3] Beizer, B. *Software Testing Techniques*. Van Nostrand Reinhold, 1990.

[4] Ericson, T., Subotic, A., and Ursing, S. TIM — a test improvement model. *Journal of Software Testing, Verification and Reliability 7*, 4 (Dec. 1997), 229–246.

[5] Foreman, L. M., and Zweben, S. H. A study of the effectiveness of control and data flow testing strategies. *The Journal of Systems and Software 21*, 3 (June 1993), 215–228.

[6] Gelperin, D. A testability maturity model. In 5^{th} *International Conference on Software Testing, Analysis and Review* (Orlando, Florida, USA, May 1996).

[7] Hamlet, D., and Taylor, R. Partition testing does not inspire confidence. *IEEE Transactions on Software Engineering 16*, 12 (Dec. 1990), 1402–1411.

[8] Hamlet, R. G. Probable correctness theory. *Information Processing Letters 25*, 1 (Apr. 1987), 17–25.

[9] Howden, W. E. Theory and practice of functional testing. *IEEE Software 2*, 5 (Sept. 1985), 6–17.

[10] Ince, D. C., and Hekmatpour, S. Empirical evaluation of random testing. *The Computer Journal 29*, 4 (Aug. 1986).

[11] Lott, C. M., and Rombach, H. D. Repeatable software engineering experiments for comparing defect detection techniques. *Empirical Software Engineering 1*, 3 (1996), 241–277.

[12] Miller, J., Roper, M., Wood, M., and Brooks, A. Towards a benchmark for the evaluation of software testing techniques. *Information and Software Technology 37*, 1 (1995), 5–13.

[13] Paulk, M. C., Curtis, B., Averill, E., Bamberger, J., Kasse, T., Konrad, M., Perdue, J., Weber, C., and Withey, J. Capability maturity model for software. Tech. Rep. CMU/SEI-91-TR-24 ADA240603, Software Engineering Institute (Carnegie Mellon University), 1991.

[14] Reid, S. C. The software testing standard — how you can use it. In 3^{rd} *European Conference on Software Testing, Analysis and Review (EuroSTAR '95)* (London, Nov. 1995).

[15] Reid, S. C. An empirical analysis of equivalence partitioning, boundary value analysis and random testing. In 4^{th} *International Software*

Metrics Symposium (Albuquerque, New Mexico, Nov. 1997), IEEE Computer Society Press, Los Alamitos, California, USA.

[16] SELBY, R. W. Combining software testing strategies: An empirical evaluation. In *Proceedings of the Workshop on Software Testing, 15–17 July, Banff, Canada* (1986), IEEE Computer Society Press, pp. 82–90.

[17] WEYUKER, E. J. The evaluation of program-based software test data adequacy criteria. *Communications of the ACM 31*, 6 (June 1988), 668–675.

[18] WEYUKER, E. J. The cost of data flow testing: An empirical study. *IEEE Transactions on Software Engineering 16*, 2 (Feb. 1990), 121–128.

[19] WOOD, M., ROPER, M., BROOKS, A., AND MILLER, J. Comparing and combining software defect detection techniques: A replicated empirical study. In *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)* (Sept. 1997), M. Jazayeri and H. Schauer, Eds., Lecture Notes in Computer Science Nr. 1013, Springer–Verlag, pp. 262–277.